ECCO Working paper, 2006-02, version: January 16, 2006

A Nodal Growth Algorithm for Concept Discovery

Francis HEYLIGHEN

ECCO, Free University of Brussels, Pleinlaan 2, B-1050 Brussels, Belgium

http://pespmc1.vub.ac.be/HEYL.html

ABSTRACT. This working paper proposes a new type of algorithm for the discovery of invariant, higher-order concepts in variable, noisy data. The algorithm is novel in that it is based on the unlimited addition of nodes and links to a recurrent connectionist network. Nodes are created to represent patterns of co-activation of existing nodes that are not sufficiently accounted for by existing nodes. This allows the system to produce novel concepts or degrees of freedom that capture the complex correlations over time and/or space of the activations of different sensory elements. The strengths of the links connecting all the nodes in the network (newly created as well as initially given) are determined according to a Hebbian or Delta learning algorithm, thus guaranteeing a network architecture that constantly adapts to its experiences. The paper sketches various potential applications, e.g. in text comprehension, clustering of scientific papers, and situated learning, that can be used to test and optimize the proposed algorithm.

Introduction

There is a fundamental unsolved problem in cognitive science, and science in general: the modelling of *creative* change; more specifically, the emergence of truly *novel concepts*. By concept, I mean one of the basic distinctions or dimensions of classification, such as light-dark, warm-cold, bird-non-bird, or alive-dead, that we use to organize our experiences. The order imposed by a concept can be discrete (either something is alive, or it is not), in which case we may call it a *distinction*, or continuous (something can be more or less warm), in which case we may call it a *dimension*. In either case, a concept determines a new variable or *degree of freedom*, since we now have become aware that phenomena can vary in this respect, e.g. being more or less warm, or either alive or dead.

A *novel* concept is a degree of freedom that cannot be reduced to a straightforward logical or mathematical combination of existing variables or distinctions. For example, we might define "purpleness" as a linear combination of "blueness" and "redness", or "volume" as the product of "width", "depth" and "height", but we cannot reduce "birdness" to a formal combination of other features: a bird is a bird, and not just a warm-blooded, egg-laying vertebrate. At best, existing scientific methodologies allow us to *decrease* the number of degrees of freedom, thus extracting the most important variables or concepts from a mass of mutually dependent variables. An example of such a method is Principal Components Analysis (a type of factor analysis), a statistical technique that allows you to derive the dimensions that are most important to explain the variation within a large amount of data.

In practice, new scientific concepts are generated purely by the scientists themselves, with very little help from formalisms or methodologies. It is clear that our brain is able to make creative inferences, coming up with concepts and constructs that put problems in a wholly different light. This often allows us to see connections between apparently disparate features and to simplify our models so that suddenly everything seems to fall into place. This ability suggests that we may be able to grasp the process of concept discovery by examing the functioning of the brain. Neural network models of brain functioning are indeed capable of impressive feats, such as learning new connections, extracting higher level patterns, and even performing the equivalent of principal components analysis [McLeod et al., 1998]. But, like all scientific formalisms until now, they can only reduce degrees of freedom, extracting the core variables from a mass of given variables; they cannot create independent variables. This is not surprising if we look at the basic components in a neural network: nodes and links. Both nodes and links are given by the designer; the network can only change the weights of the links.

In earlier research [Heylighen & Bollen, 2002], I have proposed connectionist models in which *links* can be created or destroyed, or nodes can be merged [Heylighen, 2001] making the model potentially more creative. But link creation could be seen merely as a variation of the classical connectionist model, in which links are represented as weights in the matrix of all possible connections between nodes, with “no link” being equivalent to “weight = 0”. Creating and destroying links is then not much more than allowing weight changes to pass through 0.

The present paper proposes a new paradigm that goes one step further: it moreover allows the *creation of nodes*. A node in a connectionist network essentially represents a degree of freedom, since it is characterized by its level of activation—which can either vary continuously (e.g. between 0 and 1), or discretely (e.g. switching between the binary states “on” and “off”). Since the matrix of cross-connections is defined by the nodes, changing the number of nodes necessarily makes you “jump out of the system” , i.e. it requires the definition of a new matrix.

However, unless we want nodes to be merely added at random, resulting in degrees of freedom that are meaningless because they do not connect to existing concepts, we will need a new “system” to organize the nodes. The idea is that concepts would be created in an intelligent way, with the new variables as much as possible grounded into already understood variables, while still providing an unlimited scope for novel insights. The system I propose here is based on an extension of the Hebbian algorithm, which has proved to be so powerful in directing link creation and adaptation, to the creation of nodes. In a nutshell, a node will be created to represent the *co-activation,* over space and/or time, of a cluster of existing nodes, linking into each member of the cluster with a link weight proportional to its degree of activation. Each subsequent activation of all or some of these nodes will lead to an update of the input links, strenghtening or weakening them as the case may be. As these links define the new node’s meaning or role within the network, this continuing update will make the node increasingly better at representing frequent combinations of experienced phenomena.

Note that this procedure could be viewed as an extension of Hebbian learning to *hypergraphs*. In a traditional network or graph, a link connects one node to one other node. In a hypergraph, a single link can connect a collection of nodes to a collection of other nodes. The new nodes created by the proposed algorithm could be seen as such a hyperlink, providing a bridge between a collection of frequently co-activated input nodes and a collection of output nodes. Thus, "node learning" could be seen as a form of "hyperlink learning", where the hyperlinks connect collections of existing nodes.

It must further be noted that this is certainly not the first proposal to develop a neural network that can grow nodes rather than just links [Alpaydin, 1994]. However, these earlier proposals remain within the traditional paradigm of "*feedforward*" neural networks, characterized by a linear sequence going from a given layer of input nodes via one or more layers of "hidden" nodes, to the final layer of output nodes. New nodes are typically added to the hidden layers, and can thus be seen as mere additional linear combinations of the input nodes that may facilitate the preparation of the output pattern. The present proposal assumes a freely structured *recurrent* network architecture, without a priori distinctions between layers. This allows activation to cycle without limit through the network before encountering any final "output layer", thus making the network sensitive to patterns in time and not just in space. As will be argued later, it is precisely the capturing of patterns in time that makes possible an unlimited generation of new degrees of freedom.

Let us start by formulating the underlying algorithm in more detail.

Formal definition of network dynamics

Network structure and activation

Consider a weighted, directed network $N = (V, L, W)$, where $V = \{v_1, v_2, v_3, v_4, ...\}$ is the set of nodes, $L \subset V \times V$ is the set of links between nodes and $W: L \to [0, 1]: (v_i, v_j)$

$\rightarrow w(v_i, v_j) = w_{ij}$ is the weight function with the associated weight matrix $W = (w_{ij})$. We may assume that W is normalized like a transition probability:

$$\sum_j w_{ij} = 1$$

We moreover consider an activation function A that varies over time $t \in T$ (which for simplicity we assume to be discrete):

$$A: N \times T \rightarrow [0, 1]: (v_i, t) \rightarrow A(v_i, t) = a_i (t)$$

Activation can be split up into an external and an internal component, depending on whether the activation originates outside or inside the network:

$$A(v, t) = A_{ext}(v, t) + A_{int}(v, t)$$

A_{ext} is determined by outside stimulation (e.g. nodes connected to sensors react to the sensory input by becoming activated). A_{int} is the result of *spreading activation*, i.e. the process by which activation internal to the network is passed on from node to node, proportionally to the link weights:

$$a_{int\ j} (t) = k\sum_i w_{ij}.a_i (t–1)$$

$0 < k \leq 1$ is a damping parameter, which ensures that internal activation eventually decays, so that the network would stop being active without external activation.

Link learning

Link weights will adjust to the flow of activation through the network. We can use two basic type of algorithms to control the changes, Hebbian and delta. In the *Hebbian* algorithm, link weight is incremented with a small amount each time both linked nodes are active together (*co-activation*), and this proportionally to the strength of activation in each node:

$$w_{ij} (t+1) = w_{ij}(t) + c.a_i(t).a_j(t)$$

$0 < c \leq 1$ plays here the role of a learning parameter. The higher c, the bigger the influence of the most recent activation on the weights (and therefore the weaker the influence of earlier rounds of learning). With this rule, weights can only increase but if we moreover take into account the requirement of normalization of weights, it implies that links between nodes that were not co-activated (or whose co-activation value was low) will become less strongly connected relative to those that were.

The *delta rule* takes into account the difference between external and internal activation, assuming that external activation represents the external situation as it is, whereas internally generated activation is merely an attempt to anticipate the true situation, and therefore may need to be corrected. This correction takes place by adjusting the link weight so that the internal activation it generates comes closer to the external activation it received:

$$w_{ij}\ (t+1) = w_{ij}(t) + c.\ a_i(t).\ (a_{ext\ j}(t) - a_{int\ j}(t))$$

Note that w_{ij} remains the same if internal activation equals external activation, meaning that the network perfectly anticipated. Also note that with this rule co-activation can lead both to an increase or to a decrease of the weight, thus resulting in a more fine-tuned correction.

However, this classic form of the delta rule has the shortcoming that it cannot be applied to nodes that do not receive external activation, and these are precisely the "higher order concept" nodes that we are interested in. An extension in that case would distinguish not between externally and internally generated activation, but between activation flowing through this link (internally), and the whole of activation flowing through other, indirect links (internally and/or externally). This results in the following adjustment of link weights:

$$w_{ij}\ (t+1) = w_{ij}(t) + c.\ a_i\ (t).\ (a\ _j(t) – a_i(t–1).w_{ij}(t))$$

Creation of nodes from activation clusters

At each moment t, a limited cluster $C(t) \subset V$ of nodes will be significantly activated:

$$C(t) = \{v_i \in V|\ A(v_i, t) > a_0\}$$

where $1 > a_0 \geq 0$ is a threshold parameter which we can choose so that the cluster is not too large. The nodal growth algorithm will now at each moment t add a new node c to V, and the corresponding (bidirectional) links $\{(v_i,c),(c,v_{i,}) (|\ v_i \in C\}$ to L. These new links will have the weights:

$$W(v_i, c) = W(c, v_{i,})) = A(v_i, t) = a_i\ (t)$$

The new node can be represented in vector form as a list of its input weights:

$c_i = W(v_i, c), i = 1, \dots |V|$

This is the basic algorithm which at each moment adds one node and several weighted links to the network.

Note that it is important to choose the threshold a_0 well in cases where many nodes are simultaneously activated, and where many subsequent activation patterns are encountered, because in that case the number of clusters to be considered may grow exponentially with the number of nodes. Assuming the worst case scenario that subsequent activations are independent and of unlimited extent, the total number of possible clusters, and thus of new nodes, will be of the order of $2^{|V|}$. In reality, we can assume that the great majority of potential combinations of node activations will never occur, but this will depend on the amount of regularity or redundancy in the domain, which can only be ascertained by testing the data from that domain.

To avoid an explosion in the number of potential nodes that need to be considered, we can replace the threshold a_0 by a simpler limit on the number of nodes that can make up a cluster: e.g. only consider clusters of maximum size 5. This does not necessarily limit the maximal complexity of novel concepts that can be generated in this way, as the algorithm works recursively, and thus will also consider clusters of nodes that already represent a 5-node cluster, and thus be able to generate 25-node clusters, 125-node clusters, etc. This strategy of building hierarchical complex systems by recursively combining small assemblies is a fundamental mechanism in problem-solving and evolution [Simon, 1969].

Moreover, in order not to let the network grow too big, adding lots of nodes that are mostly the same in their linking pattern, we should add the restriction that the node to be created should be sufficient *novel*. The idea is that nodes only need to be created to capture configurations of stimuli that cannot be reduced to already existing linking patterns. To quantify this requirement, we can use the cosine similarity measure between the activation vector ($c_i'(t')$) representing the new cluster of activation $C(t' > t)$ and the one ($c_i(t)$) that had been captured earlier:

$$sim(c,c') = \frac{\sum_i c_i c'_i}{\sqrt{\sum_i c_i^2 \sum_i c_i'^2}}$$

Note that sim(x,y) = 1 iff x and y are equal (or proportional), and sim(x,y) = 0 iff x and y are orthogonal. If we now define a similarity threshold $0 < s < 1$, then the novelty condition for node creation can be formulated as:

if $\exists$ c such that $sim(c,c') > s$, then do not create a new node c'

A quick way to check this condition is to let activation spread from the activation pattern c' for one step. The node that gets the highest activation is the one most similar to the potential new node, and therefore we only need to check for this node whether the similarity remains below the threshold. This leads us to another interpretation of the novelty condition: new nodes are created whenever there is not any single existing node that can satisfactorily *account* for the activation, in the sense that it will attract or absorb a significant part of that activation in a single step.

To further avoid combinatorial explosion, we must put a final restriction on the number of newly created nodes. After the creation of a node c, the system may encounter hardly any activation pattern able to activate its input links sufficiently for it to become significantly activated. In that case, Hebbian or delta learning will gradually weaken the incoming links (v_i, c) , until the node hardly receives any input at all. Once the total incoming strength has fallen below a certain threshold $0 < b << 1$, we may decide to eliminate the node and its corresponding links:

if $\sum_i W(v_i, c) < b$, then erase node c

This mechanism will regularly prune the network to eliminate spurious clusters of nodes that may have appeared once, but don't have any predictive power. For example, consider a system learning new words by creating nodes for the combinations of letters it encounters. Some of these combinations (e.g. misspellings) will be meaningless, and therefore should be erased if nothing similar to them is encountered in a long while.

Further development of new nodes

As soon as a new node c is created, its links will start adapting, according to the Hebbian or delta algorithms. This means that when c is (significantly) activated together with some other node v, the links (v, c) and (c, v) will be strengthened. Simultaneously, links with nodes that are not simultaneously active will weaken and eventually disappear. This means that the linking pattern of c will increasingly start to reflect the most common patterns of activation.

Suppose that our network represents the brain of a young child which for the first time sees a bird, say a robin, which eats some grains and then flies away. Already known features, such as "has two legs", "flies", "eats grains", "has a red breast", "has two eyes", "is small", etc. will be activated. Assume that none of the already existing concepts, such as "dog", "person" or "toy", is similar, i.e. has largely overlapping features. The child will therefore create a new concept node, which we will call "bird". This concept is defined by all and only the features of that first perception, with the same relative strength as the impression they made at that moment.

Suppose that a little later, the child perceives another bird, say a blackbird eating a worm. Many of the same features will be activated, e.g. "two legs", "flies", "small", ... This means that there is sufficient similarity with the previous experience to map these on the same concept, rather than creating a new concept node. But the similarity is not perfect: "red breast" and "eats grains" are no longer activated, "small" has less activation, and "black" and "eats worms" are this time activated. Because of learning, the linking pattern defining the bird concept will now adjust to take these differences into account: the connections with "flies", ... are strengthened, those with "red breast", ... are weakened, while connections to "black", ... are added. What remains is a linking pattern that neither fully represents a blackbird, nor a robin, but that is somehow an average of the two.

For every further encounter with something bird-like, the concept will in the same way continue to adjust itself, weakening its links to rare or spurious properties, such as "red breast", that appear only once or twice, while strengthening its links to essential or typical properties, such as "two legs" or "flies", that are encountered again and again. After a while, the linking pattern of the new "bird" node will represent a *prototypical* bird, with strong links to the most common bird properties and increasingly weaker links to the less frequent ones. As a result, the child will very quickly learn to recognize typical birds, since the activation of just a few strong links such as "flies" or "feathers" will be enough for it to activate (and thus be prepared to perceive) all the other typical bird features, such as "beak" or "lays eggs". But it may still have difficulty recognizing non-typical birds such as penguins or ostriches, until it develops a specialized "penguin" node that is linked to "bird", but still has some peculiar links of its own, such as "swims", "cannot fly", ...

Note that the creation of nodes is *recursive*: a new node will be defined by its pattern of links to whichever nodes were active at the moments of its creation, including nodes that were themselves created in this way at an earlier stage. Thus, the "penguin" node, if is is created after the bird node, is likely to be strongly connected to that node, and thus inherit many, but not all of its properties. This mechanism also allows the creation of hierarchies of nodes, producing levels of complexity. We may define the primitive, "sensory" nodes that define the initial network as belonging to level 0 of the hierarchy. A new node, whose input nodes consist (primarily) of level 0 nodes then belongs to level 1. Recursively, a node whose input nodes belong primarily to level n can be defined as belonging to level n+1.

Higher level nodes will become increasingly abstract and complex, presupposing a whole complex of earlier, simpler concepts in order for them to be recognized. Higher order nodes will also be more stable or *invariant* [Hawkins, 2004; George & Hawkins, 2005; Booth & Rolls, 1998], in the sense that since they get activation directly or indirectly from many different input sources, they can remain activated even when the actual input is very fragmented and variable. For example, a bird hiding in a bush so that one moment only a beak and a leg are visible, the next moment only some feathers and a tail, will be still be recognized throughout all the perceptual changes as a bird, since each of these elements has built up a strong connection to the overall "bird" concept, so that only a few of them are necessary to activate the concept.

Also note that nodes do not need to have an unambiguous position in a hierarchy, as their incoming links may crosscut with many different levels of the hierarchy, and as their activation will propagate as well "upwards", "downwards", as "horizontally", thus allowing complex feedback loops. Therefore, this model is much more general and flexible than the traditional feedforward neural network representations, in which activation flows subsequently from the input layer via one or more “hidden” layers to the final output layer.

Origin of activation patterns

The present modelling methodology assumes that the network learns nodes and links from the variations in its activation function. Intuitively, the external activation function represents sensory stimuli, i.e. information about the world outside of the network which is picked up by some kind of sensory apparatus and translated into activation values for level 0 nodes. The function of the network is to become increasingly competent in anticipating these activation patterns, i.e. given a history of external activations, $A_{ext}(v, t)$, $A_{ext}(v, t-1)$, ... $A_{ext}(v, t-k)$, it should be able to generate internal activation patterns $A_{int}(v, t+1)$, $A_{int}(v, t+2)$, ..., that are as similar as possible to the external activations that will effectively follow. To achieve that, we need to assume that the outside world is to some degree regular or ordered, although its order may only become apparent at a high level of complexity, where patterns of patterns of patterns, distributed over space and time, are made visible by a recursive method of analysis and synthesis like the one we proposed. In other words, we assume that the world is complex, but not random.

The present paper claims that the nodal growth algorithm as we have sketched is potentially much more effective at such multilevel prediction of complex patterns than traditional connectionist methods, which are based on the adjustment of link weights between predefined layers of nodes. To test this hypothesis, we will need concrete data—in the form of non-trivial activation patterns—for the method to analyse. While the example of a child learning concepts by abstracting from the complex sensory stimulation that it undergoes may be very intuitive, in the present state of science we cannot hope to get such a rich array of activation patterns. An obvious alternative is to use the sensory input stream of an autonomous robot, but this is limited by various engineering constraints that we at present would rather not be concerned with. A more controllable alternative is concept learning by an agent in a simulated environment

[Gershenson, 2004], but here it is quite difficult to produce an environment of a sufficiently realistic complexity.

However, the present methodology does not make any assumptions about the environment or "agent-like" properties of the program that uses a nodal growth algorithm. That allows us to apply it to various more abstract data streams. We will classify those according to whether the data arrive sequentially or in parallel.

Sequential activation

In sequential data streams, we assume that one node is activated at a time.

An example would be an acoustic signal, where the air pressure sensed by the ear drum changes over time. This signal is one-dimensional, and can be represented by a single node whose activation corresponds to the strength of pressure. Yet, we know that sound is typically experienced and conceptualized according to many dimensions: pitch, volume, harmony, timbre, rhythm, etc. This illustrates the creation of new degrees of freedom by the neural system [Cariani, 1997]. Mathematically, the explanation is simple: any real function of one variable (e.g pressure as a function of time) can be represented as a point in an infinite dimensional space by means of Fourier or harmonic analysis. The core idea is that it is the variations over time that add variability and thus degrees of freedom to something that, considered statically, only has a single degree of freedom. However, sound, being continuous, is not easy to analyse by means of our discrete nodal growth algorithm.

Another simple example of a one-dimensional signal that is decoded as multidimensional is the electromagnetic wave that carries TV programs (or even simpler the binary sequence with which a digital TV program is coded). The wave/code is one-dimensional, with only intensity varying over time, but the resulting program has two spatial dimensions, three color dimensions, a dimension of movement and several sound dimensions. The decoding of these dimensions is preprogrammed in the TV apparatus, though, and what interests us is a system that automatically learns to create new dimensions when needed.

A discrete example of a sequential data stream is text. Text consists of a digital sequence of characters (letters, numbers, punctuation marks, ...). The symbols are chosen from a discrete alphabet. In the case of the ASCII coding, that alphabet contains 128 basic characters. Let us represent each character by a node in our network N. A text can then be represented by an activation function: A(v,t) which is such that at each moment t, there is one node x for which A(x,t) = 1, while A(y≠x, t) = 0. x represents the character that is being "read" at time t. With such an activation function, our learning algorithms won't be of much help, though, since they require that different nodes (characters) be activated simultaneously. We can tackle that problem in two ways:

1) we can let activation of a given node decay slowly so that when A(x,t) = 1, then A(x,t+1) = 0.9, A(x,t+2) = 0.81, etc. The word "cat", at the moment the last letter is

read, would then be represented by the activation pattern A("t",t) = 1, A("a",t) = 0.9, A("c",t) = 0.81. This is probably not a very reliable representation, since the word "act" would be represented by almost the same patterns, except that "c" would now be a little more activated than "a".

2) a richer representation would associate each character's reading with a time or position stamp, thus having different nodes for "c in first position" ("c", 1) and "c in second position" ("c", 2). Suppose our network distinguishes 100 positions, then we need 100 × 128 = 12800 nodes in our network.

With such a network, co-activation now becomes meaningful, and as more text is "read", the network will first learn to expect that certain letters tend to follow or not follow each other, e.g. that "c" may well be followed by "o", or "h", but not by "x" or "p". This is simple link learning. However, node learning should soon learn to recognize common morphemes, words, and phrases. These will modulate the anticipation patterns defined by the links between level 0 character nodes, so that letters can be predicted with better accuracy in a context where preceding, higher level words and phrases have already been recognized. First, the network will create strong nodes that represent very common words such as “the” or “in” followed by less common but still regularly recurring words such as “information”, and common phrases such as “by the way”, “this is”, “do you know”, ... The latter can be formed either directly from input activation patterns on the character nodes, or from derived activation patterns on previously learned word nodes.

Parallel activation

In the simplest case, we can have co-activations of several nodes simultaneously, but without any meaningful temporal ordering. This is the case with *co-occurrence* data: we know that concepts or nodes a, b, c, d... often occur together, but these occurrences are static, without any arrow of time. For example, we may have a list of papers together with their keywords. Suppose each keyword is represented by a node. Each paper can then be represented as an activation pattern, activating all those keywords it contains. However, the temporal ordering of these papers is irrelevant, as there will be no meaningful correlation between paper n and paper n+1 where n represents their order in the database or listing (e.g. alphabetical).

This is no problem for the Hebbian or nodal growth algorithm. Whatever the order in which the activation patterns are presented, the algorithm will create or strengthen links and nodes whenever the same keywords appear together. In that way, the network should be able not only to predict which keywords are likely to follow, but to create clusters of commonly co-occurring keywords that perhaps define research subjects, communities and disciplines, thus producing a taxonomical ordering of papers and keywords. This means that if you activate the network with a few keywords, it will generate a list of additional keywords that you can expect to find at the level of papers, collections, topics or communities of papers.

The roles of paper and keyword can be switched in this representation, with papers corresponding to nodes, and keywords activating all papers that contain the keyword. After the network has been trained, the activation of a few papers should lead to the activation of the “higher order” nodes representing the keywords (or families of commonly co-occurring keywords) that these papers have in common.

Many other examples can be given of such co-occurrence patterns that could be meaningfully analysed by our nodal growth algorithm: e.g. books bought or borrowed by the same customers, features shared by different animals or plants, DNA sequences shared by the same kinds of organisms, etc.

Parallel and sequential activation

In most real processes we will get information both in parallel and sequentially. E.g. during visual perception several features are perceived simultaneously, but these features moreover change over time. Another example would be science dynamics, where we can consider not only the co-occurrence of keywords or papers, but the evolution of these occurrences over time as we compare papers published in subsequent years. For example, we may find that a particular new research community, characterized by a cluster of keywords and key papers, gradually emerged over a period of a dozen years, only to split up into several more specialized communities in the following decade.

An advanced version of the nodal growth algorithm may first learn to distinguish relatively static clusters of co-occuring phenomena, but then start to learn their dynamics as it notes that certain patterns tend to be followed by specific other patterns.

Testing the algorithm

Up till now, we have presented the algorithm in a purely conceptual way, arguing with the help of examples and general principles that it may potentially discover very complex patterns in situations where other machine learning or data mining algorithms are likely to get stuck. However, it is well possible that we have overlooked crucial difficulties that may keep the algorithm from functioning in practice. Therefore, we need to test it concretely, in a situation that is sufficiently complex and realistic to convince observers that the algorithm has some real advantages over others.

The parallel activation or co-occurrence case is probably the simplest, as it covers a whole range of situations that have already been extensively explored by data mining algorithms, such as various forms of clustering algorithms. This would allow a direct comparison with the results of these algorithms. However, the problem is that the quality of clustering is intrinsically difficult to estimate, as there is no "objective" measure of what is a good cluster. Moreover, different algorithms will typically perform well on different types of data, and being successful in one domain gives little guarantee of success in another domain.

A more convincing test might be carried out in the sequential case, and more specifically in the learning of text. Text has the advantage that it has a very clear low level structure (letters, words), but a very complex high level structure (phrases, sentences, contexts, stories, ...). Therefore, it is easy to start the analysis by "reading" characters and words, but very difficult to carry it through to the higher levels of textual comprehension. Thus, text forms an elegant challenge for a learning network as we sketched it, allowing it to start simple and become increasingly sophisticated as it processes more and more text. However, since there is no clear, objective measure of textual comprehension, we need another way to measure its performance.

A general characteristic of any "intelligent" system is its capability for prediction or anticipation [Hawkins, 2004; George & Hawkins, 2005]. The idea is that the more text the network has assimilated, the better it should become at anticipating as yet unseen texts. This prediction capability can be measured very easily in the following way: from an existing text, you erase k % of the characters and replace them by question marks (e.g. k= 50 means removing half of the characters). A good network, like our own brain, should be good at guessing what the erased characters are, by using the surrounding characters that were not erased to guess the morphemes, words, phrases or contexts to which they belong. Assume that we require a given level of accuracy in the guessing (e.g. at least 95% of the guesses should be correct). In that case the quality of the network can be measured by the highest number k of characters that can be randomly removed before the guesses fall below the required level of accuracy. This allows us to compare various networks (e.g. that have been trained using variable amounts of text, or using different learning parameters) with other machine learning systems (e.g. Hidden Markov models) and even with human subjects.

Conclusion

We have presented a promising new type of algorithm for the discovery of invariant, higher-order concepts within a mass of variable, noisy data. The algorithm is novel in that it is based on the unlimited creation of new nodes within a recurrent connectionist network. Nodes are created to represent patterns of co-activation of existing nodes that are not sufficiently accounted for by existing nodes. This allows the system to produce novel concepts or degrees of freedom that capture the complex correlations over time and/or space of the activations of different sensory elements. The strengths of the links connecting all the nodes in the network (newly created as well as initially given) are determined according to a Hebbian or Delta learning algorithm, thus guaranteeing a network architecture that constantly adapts to its experiences.

The proposed algorithm can in principle be applied to any task involving learning, discovery or data mining. We have sketched some simple applications in text comprehension of clustering of papers and keywords, and suggested that it is equally applicable to the more complex spatio-temporal information streams encountered by situated agents. The next step will be to actually implement and test the algorithm on one of these types of data, preferably comparing the quality of its predictions with those of other machine learning algorithms.

References

Alpaydin E (1994) "GAL: Networks that Grow when they Learn and Shrink when they Forget'" International Journal of Pattern Recognition and Artificial Intelligence , 8, 391-414.

Booth, M.C., Rolls, E.T., "View-invariant representations of familiar objects by neurons in the inferior temporal visual cortex", *Cerebral Cortex*, volume 8(6), pp. 510-523, 1998.

Cariani, P. (1997) Temporal coding of sensory information. In (J. M. Bower, Ed.) Computational neuroscience: Trends in Research 1997 (Plenum), pp. 591-598.

Cariani, P. (1997) Emergence of new signal-primitives in neural systems, Intellectica, 19972, p.95-143.

George, D., Hawkins, J., "A Hierarchical Bayesian Model of Invariant Pattern Recognition in the Visual Cortex", *Proceedings of the International Joint Conference on Neural Network*s, IEEE, 2005.

George, D., Hawkins, J., "Invariant Pattern Recognition using Bayesian Inference on Hierarchical Sequences", *Redwood Neuroscience Institute Technical Report*, 2005.

Gershenson, C. (2004). Cognitive Paradigms: Which One is the Best? *Cognitive Systems Research*, 5(2):135-156, June 2004.

Hawkins, J., Blakeslee, S., *On Intelligence: How a New Understanding of the Brain will lead to Truly Intelligent Machines*, Henry Holt and Company, 2004.

Heylighen F. & Bollen J. (2002): "Hebbian Algorithms for a Digital Library Recommendation System", in Proceedings 2002 International Conference on Parallel Processing Workshops (IEEE Computer Society Press)

Heylighen F. & C. Gershenson (2003): "The Meaning of Self-organization in Computing", IEEE Intelligent Systems 18:4, p. 72-75.

Heylighen F. (2001): "Bootstrapping knowledge representations: from entailment meshes via semantic nets to learning webs", Kybernetes 30 (5/6), p. 691-722.

McLeod, P., Plunkett, K. & Rolls, E. T. (1998). *Introduction to connectionist modeling of cognitive processes*. Oxford, UK: Oxford University Press.